Virus Simulation Group

Software Reuse

In our project we used many levels of reused of code as a starting point that helped create the finished product. Considering our project from an abstract level we used our pervious knowledge of our use of unity and the creation of circular arrays and other pre-implemented data structures that were instrumental to our creation of the pathfinding AI that we used in our program. For the object level we took advantage of many of the libraries inside unity that helped decrease time to develop these systems from scratch, like our A star pathfinding system. In the component level of reuse we reused code from a past unity project. We used specific parts of the project like movement and parts of the pathfinding. Of course, it had to be modified extensively for use within our project. At the system level there was no reuse as we did not have access to another kind of application of the same type and thus most of our system structure is from scratch. Thus, our destinations and data structure accessors and structuring the animation system were mostly done from scratch, with some assistance from physics software. We gained a head start in our program from software reuse that gave us a basis to expand from which not only saved time but gave us a direction to go from which was extremely helpful.

The cost of time on our program was significantly altered due to the fact that we ended up reusing commonly used software models in terms of physics programming. For example, one of our members already had a character programmed for a game project on the side. We reused the controls he had programmed into his game and used them for the player in our project. This cut down on the amount of time we had to spend on the character controls allowing us to move onto other portions of the program such as the city and virus spreading. Another important part

that helped with the cost was the prior knowledge of unity several members have. Thanks to the prior knowledge we were easily able to slide into using the program and understand what was required to make the project. This again cut down on the cost of time to learn how to use the program. On top of this, we reused particle systems and real-time rendering effects that were used in previous menu interfaces that some members of the group had implemented. All of these fall into the component level of reuse.

With the end of the project coming up, if we do decide to change anything if we started from scratch we would try to find code that could let us better implement the menu system. For example, we went through several movement systems, as the collisions and physics systems didn't interact well with user defined raycasting. It also took a bit of research into using outsourced software to use alpha-channels and transparency levels on some of our art and animation systems. But one universal aspect that came into play for most of the project was previous programming knowledge and practice. Without the practice and level of comfort afforded by classes and years of algorithmic programming experience in languages such as Java and C#, this project would be light-years behind what it is. As such, much of our project is a testament not only to hours of hard work and research into the systems of reusable code in Unity, but also to many of the teachers we have had in the past who enforced a policy of critical thinking and abstract mathematical know-how. All-in-all, we have had many reusable assets contribute to our project, and it has been an important lesson in originality and utilizing all the resources at our disposal to create the best, working, and efficient software project possible.